Title	Weakly nonlinear circuit analysis based on fast multidimensional inverse Laplace transform
Author(s)	Wang, T; Liu, H; Wang, Y; Wong, N
Citation	The 17th Asia and South Pacific Design Automation Conference (ASP-DAC 2012), Sydney, Australia, 30 January-2 February 2012. In Asia and South Pacific Design Automation Conference Proceedings, 2012, p. 547-552
Issued Date	2012
URL	http://hdl.handle.net/10722/158785
Rights	Asia and South Pacific Design Automation Conference Proceedings. Copyright © IEEE.

Weakly Nonlinear Circuit Analysis Based on Fast Multidimensional Inverse Laplace Transform

Tingting Wang, Haotian Liu, Yuanzhe Wang, and Ngai Wong

Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam Road Hong Kong
e-mail: {ttwang, htliu, yzwang, nwong}@eee.hku.hk

Abstract— There have been continuing thrusts in developing efficient modeling techniques for circuit simulation. However, most circuit simulation methods are time-domain solvers. In this paper we propose a frequency-domain simulation method based on Laguerre function expansion. The proposed method handles both linear and nonlinear circuits. The Laguerre method can invert multidimensional Laplace transform efficiently with a high accuracy, which is a key step of the proposed method. Besides, an adaptive mesh refinement (AMR) technique is developed and its parallel implementation is introduced to speed up the computation. Numerical examples show that our proposed method can accurately simulate large circuits while enjoying low computation complexity.

I. Introduction

In recent years, numerical inverse Laplace transform has become recognized as an important tool in many disciplines, such as computation of probability distributions in stochastic models, analysis of time-dependent performance of stationary and non-stationary systems [1], solving multi-time partial differential equation systems in frequency domain [2, 3] and weakly nonlinear circuits analysis [4]. However, most existing work on numerical inverse of Laplace transform handles simple cases with only up to two dimensions, and their computation is prohibitively expensive when applied to higher-dimensional cases [4, 5]. Numerically sound and efficient algorithm to compute the inverse Laplace transform of higher dimensions remains to be developed.

On the other hand, there have been continuing thrusts in developing efficient modeling techniques for simulating the behaviors of nonlinear analog or radio-frequency (RF) blocks in modern mixed-signal chips [6]. There are many routines for simulating nonlinear systems. Time-domain solvers are preferred for highly nonlinear circuits. However, they are not suitable for multi-tone problems. Frequency-domain schemes can handle multi-tone problems [2], however it still requires a multi-dimensional transform inverse process or a time-consuming association of variables [7]. Therefore an efficient numerical multidimensional inverse Laplace transform technique is in great need for simulation of nonlinear circuits in the frequency domain.

In this paper, we present an effective algorithm for the fast numerical multidimensional inverse Laplace transform, as well as its application to weakly nonlinear circuit simulation. Our method is based on Laguerre function expansion. A fast Fourier transform (FFT)-based extension of the variant of the Fourier-series method is used to speed up the coefficient computation of the multidimensional Laguerre generating function [1]. An adaptive mesh refinement (AMR) method is proposed to speed up computing the multidimensional inverse Laplace transform. To further utilize the Laguerre method in calculating numerical Laplace inverse, especially that of high dimensions, we propose an effective parallel implementation based on both quadcore CPU workstation and the graphic processing unit (GPU) platform, which can further speed up the computation. The purpose of this paper is to show that Laguerre-based numerical inverse Laplace transform can achieve fast speedup even in high dimensions. The method is applied to simulate two nonlinear circuit examples and accurate results are obtained to show the effectiveness of our approach.

The main contributions of this paper include: 1) we extend the existing Laguerre-based 2-dimensional inverse Laplace transform procedure to an N-dimensional one; 2) a parallel routine is developed for computing the numerical inverse of multidimensional Laplace transforms, the method can speed up the computation by as much as more than 3 times on a quadcore CPU, and much more on a GPU platform; 3) AMR is used to achieve further speedup. AMR, when implemented in parallel, serves as a fast method for computing high-dimensional numerical inverse Laplace transform, where traditional methods are extremely inefficient if not impossible. The proposed method is able to rapidly invert high-order Laplace transforms, which in turn provides an efficient way to simulate large scale nonlinear circuits with accurate results.

This paper is organized as follows. Section II extends the basic FFT-based Laguerre method used in the numerical inverse of Laplace transform to N-dimensional inverse. In Section III, an AMR technique and a parallel implementation are proposed to speed up the computation. The results of the parallel and non-parallel routines are compared. Then, in Section IV, the proposed method is applied to solve a simple weakly nonlinear circuit and a large scale nonlinear circuit. Section V concludes our work.

978-1-4673-0772-7/12/$31.00 ©2012 IEEE

II. The Laguerre-based Numerical Multidimensional Inverse Laplace Transform

A. Background on Numerical Inverse Laplace Transform

The multidimensional Laplace transform of a real function $f(t_1, t_2, \ldots, t_n)$ of n variables t_i , $i = 1, 2, \ldots, n$ is defined as

$$F(s_1, s_2, \ldots, s_n) = \int_0^\infty \int_0^\infty \cdots \int_0^\infty f(t_1, t_2, \cdots, t_n) \cdot e^{-s_1 t_1 - s_2 t_2 \cdots - s_n t_n} dt_1 dt_2 \cdots dt_n. \quad (1)$$

The problem of numerical inverse for the Laplace transform $F(s_1, s_2, \ldots, s_n)$ is to determine the approximations for $f(t_1, t_2, \ldots, t_n)$ when the numerical values of the multidimensional function $F(s_1, s_2, \ldots, s_n)$ are known.

There have been many algorithms for numerically inverting Laplace transforms in one and two dimensions. These algorithms include the following approaches: 1) Fourier series expansion; 2) Laguerre function expansion; 3) Deforming the Bromwich contour; 4) Combination of Gaver functionals [8]. The Laguerre function expansion has been used in numerical inverse Laplace transforms for decades, and [9] has shown that for well-behaved functions, the Laguerre method can be more efficient than other methods, especially when many function values are required. Through several improvement such as a scaling and summation technique [1], the Laguerre method can also work efficiently for a larger range of functions. Our work is based on the Laguerre function expansion, which is briefly introduced below.

B. Laguerre-Series Representation

We extend the 2-dimensional inverse Laplace transform in [1] to a general N-dimensional case. The numerical inverse of Laplace transform can be implemented based on the Laguerre-series representation. A multi-variable function can be represented as a weighed sum of N-variable Laguerre functions, as shown below

$$f(t_1, t_2, \ldots, t_N) = \sum_{n_1=0}^\infty \sum_{n_2=0}^\infty \cdots \sum_{n_N=0}^\infty q_{n_1,n_2,\ldots,n_N} \cdot l_{n_1}(t_1) l_{n_2}(t_2) \cdots l_{n_N}(t_N), \quad (2)$$

where

$$l_{n_i}(t) = e^{-t/2} L_{n_i}(t), \quad t \geq 0, \quad (3)$$

$$L_{n_i}(t) = \sum_{k=0}^{n} \binom{n}{k} \frac{(-t)^k}{k!}, \quad t \geq 0, \quad (4)$$

with $q_{n_1,n_2,\ldots,n_N}$ being the Laguerre coefficients, $l_{n_i}(t)$ the associated Laguerre functions, $L_{n_i}(t)$ the Laguerre polynomials. The Laguerre functions $l_{n_i}(t)$ form a set of orthogonal bases for the Laplace transform.

We show below the relationship between the Laguerre coefficients $q_{n_1,n_2,\ldots,n_N}$ and the Laplace transform.

The Laplace transform of the Laguerre function $l_n(t)$ is

$$\hat{l}_n(s) = \int_0^\infty e^{-st} l_n(t) dt = \frac{2(2s-1)^n}{(2s+1)^{n+1}}. \quad (5)$$

Now perform Laplace transform on both sides of (2), we get

$$\hat{f}(s_1, s_2, \cdots, s_N) = 2^N \sum_{n_1=0}^\infty \cdots \sum_{n_N=0}^\infty q_{n_1,\ldots,n_N} \cdot \frac{(2s_1-1)^{n_1}}{(2s_1+1)^{n_1+1}} \cdots \frac{(2s_N-1)^{n_N}}{(2s_N+1)^{n_N+1}}. \quad (6)$$

By using the bilinear transformation

$$z = T(s) = \frac{2s-1}{2s+1}, \quad s = T^{-1}(z) = \frac{1+z}{2(1-z)},$$

we obtain

$$Q(z_1, \ldots, z_N) = \sum_{n_1=0}^\infty \cdots \sum_{n_N=0}^\infty q_{n_1,\ldots,n_N} z_1^{n_1} \cdots z_N^{n_N} = (1-z_1)^{-1} \cdots (1-z_N)^{-1} \hat{f}\left(\frac{1+z_1}{2(1-z_1)}, \ldots, \frac{1+z_N}{2(1-z_N)}\right), \quad (7)$$

where $Q(z_1, \ldots, z_N)$ is called the Laguerre generating function. The Laguerre generating functions can be used to calculate the Lagueree coefficients, and then we can obtain the numerical values of the original function in (2).

C. Calculating the Laguerre Functions and Laguerre Coefficients through an FFT-based Method

To get the numerical value of the original function, we first need to compute the Laguerre functions and Laguerre coefficients. The Laguerre functions $l_n(t)$ can be calculated in a numerically stable recursion way

$$l_n(t) = \left(\frac{2n-1-t}{n}\right) l_{n-1}(t) - \left(\frac{n-1}{n}\right) l_{n-2}(t), \quad (8)$$

starting with $l_0(t) = e^{-t/2}$ and $l_1(t) = (1-t)e^{-t/2}$.

Next, we calculate the Laguerre coefficients. Rewriting Eq.(3.4) in [10], we can get

$$q_{n_1,\ldots,n_N} \approx \bar{q}_{n_1,\ldots,n_N} = \frac{1}{m_1 r_1^{n_1} \cdots m_N r_N^{n_N}} \cdot \sum_{k_1=0}^{m_1-1} \cdots \sum_{k_N=0}^{m_N-1} e^{-2\pi i k_1 n_1 - \cdots - 2\pi i k_N n_N} \cdot \hat{Q}(r_1 e^{2\pi i k_1/m_1}, \cdots, r_N e^{2\pi i k_N/m_N}), \quad (9)$$

where $m_i = 2l_i n_i$, l_i is a roundoff error control parameter and r_i is a real number with $r_i < 1$ for $i = 1, 2, \ldots, N$ [1, 10]. To reduce the roundoff error one can increase the parameters l_i. Typically, $l_i = 1$ or 2 is a good choice for accuracy control. The above choice with m_j changing with n_j is appropriate if we need inversion at only a few points. However, in many practical problems, $q_{n_1,n_2,\ldots,n_N}$ has to be computed for all n_i in the range $0 \leq n_i \leq N_i - 1$, i.e., at a total of $N_1 N_2 \cdots N_N$ points, where N_i have to be a sufficiently large number to guarantee high numerical accuracy.

In our algorithm, we obtain a better choice of m_i independent of n_i. Specifically, we choose $m_i = 2l_i N_i$ for all n_i , in the range $0 \leq n_i \leq N_i - 1$.

From (9) we can define

$$a_{n_1,\ldots,n_N} = \bar{q}_{n_1,\ldots,n_N} r_1^{n_1} \cdots r_N^{n_N}, \quad (10)$$

$$b_{n_1,\ldots,n_N} = Q(r_1 e^{2\pi i n_1/m_1}, \ldots, r_N e^{2\pi i n_N/m_N}), \quad (11)$$

$$a_{n_1,n_2,\cdots,n_N} = \frac{1}{m_1 m_2 \cdots m_N} \cdot \sum_{j_1=0}^{m_1-1} \cdots \sum_{j_N=0}^{m_N-1} e^{-2\pi i j_1 n_1/m_1 - \cdots - 2\pi i j_N n_N/m_N} b_{j_1,j_2,\ldots j_N} \quad (12)$$

From (7) and (11) we obtain $b_{n_1,\ldots,n_N}$, then the N-dimensional FFT algorithm can be used to calculate $a_{n_1,\ldots,n_2}$ from (12). Specifically iterative one-dimensional "decimation in frequency" algorithms as in [1] are used. After that we can compute the desired function values $f(t_1, t_2, \ldots, t_n)$ from (2).

III. AMR and Parallelized Inverse Laplace Transform

A. A Parallel Routine for Computation

An advantage of the Laguerre method is that it can be implemented in parallel. In the Laguerre-based inverse Laplace transform computation, we calculate values at each sampling point independently. Therefore we can make use of this advantage and compute values at these sampling points simultaneously on a multi-core platform. This is an advantage over other numerical inverse Laplace transform methods such as the block-pulse method [5] , which cannot be parallelized due to the inter-correlation among time sampling points.

In this test example, the MATLAB Parallel Computing Toolbox and AccelerEyes Jacket are used to implement the CPU and GPU parallel computation. The proposed algorithm may also be implemented in other languages (such as C++ or Python).

The parallel flow is tested on a quadcore workstation as well as GPU with 480 Compute Unified Device Architecture (CUDA) cores. Example 1 is for calculating the 1D inverse of Laplace transform function $F_1(s) = \frac{1}{s}$, Example 2 is for the 2D inverse of $F_2(s_1, s_2) = \frac{1}{s_1^2 s_2 (s_2+1)}$, and Example 3 shows the 3D inverse of $F_3(s_1, s_2, s_3) = \frac{1}{s_3^2 (s_1+1)(s_2+2)(s_2+3)}$. Table I compares the computation times of the three examples computed with one-core CPU (1CPU), quadcore CPU (4CPU) and GPU with various numbers of time sampling points (time steps). Table II shows relative errors in our numerical scheme compared with that from the analytical solution. The numerical results show that the parallel routine can perform the numerical inverse with high accuracy and significant speedup. Computation based on quadcore CPU can speed up the calculation by up to 3.4 times, while GPU can provide a speedup of more than ten times.

TABLE I
Computation Times of 1D, 2D and 3D Examples computed with One-core CPU (1CPU), Quadcore CPU (4CPU) and GPU with various numbers of time steps

	Steps	1CPU	4CPU	GPU
1D	10000	0.851 s	0.306 s	0.0501 s
example	50000	4.12 s	1.199	0.0830 s
2D	900	2.03 s	0.631 s	0.139 s
example	6400	11.1 s	3.69 s	0.739s
3D	125	40.6 s	13.4 s	2.61 s
example	1000	332 s	121 s	20.9 s

TABLE II
Relative Errors of 1D, 2D and 3D Examples Computed with One-core CPU (1CPU), Quadcore CPU (4CPU) and GPU with Various Numbers of Time Steps

	Steps	1CPU error	4CPU error	GPU error
1D	10000	7.11e-13%	7.21e-13%	4.79e-13%
example	50000	7.02e-13%	7.01e-13%	4.87e-13%
2D	900	8.61e-11%	8.59e-11%	1.06e-6%
example	6400	7.23e-11%	7.24e-11%	1.10e-6%
3D	125	7.27e-6%	7.25e-6%	7.27e-6%
example	1000	6.10e-6%	5.97e-6%	6.18e-6%

B. AMR

Spacing of time grids determines the accuracy and the cost of the computation. For example, regions with steep gradients need fine grid spacing. Uniformly fine meshes are computationally costly. Therefore, adaptive methods are necessary in such simulation.

AMR [11] can be used to accelerate the computation of the Laguerre-based inverse Laplace transform. AMR starts with a base coarse time grid. Then, sub-grids are drawn recursively on the regions with fast changing slopes. Local truncation error can be used to estimate where should be refined. Taking first-order approximation, the local truncation error $E(h)$ can be expressed by the step size h and the computed results at point $x, x+h$ and $x+2h$,

$$E(h) \approx \frac{1}{2h}\left(-f(x) + 2f(x+h) - f(x+2h)\right).$$

If $|E(h)|$ is larger than a specific threshold value, say, ϵ, then smaller step size $\frac{h}{2}$ is needed in this region and the results at points $x + \frac{h}{2}$ and $x + \frac{3h}{2}$ are to be calculated. This method can be recursively applied on the simulation until all the local truncation errors are below the threshold or the maximum number of iterations is reached.

AMR can be extended to high-dimension grids easily. Furthermore, AMR of the Laguerre-based inverse Laplace transform could enjoy higher accuracy than time-domain methods since the error will not be accumulated through the process of computing. Unlike other methods, the process of AMR also can be parallelized since the calculation is independent of previous results. Numerical example of this technique will be shown in Section IV.

IV. Application to Weakly Nonlinear Circuit Simulation

Example 1

Our proposed parallel computation of multidimensional inverse Laplace transform can be applied to simulate nonlinear circuits. We first consider a simple single-input multi-output

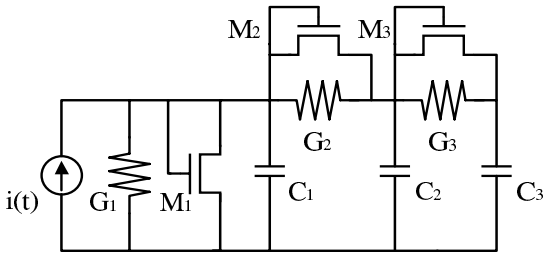

Fig. 1. A circuit with nonlinear elements.

(SIMO) system as in Fig. 1 to demonstrate how the Lagurre method can be used to simulate a weakly nonlinear circuit. Large scale nonlinear circuits can be simulated in a similar way. This SIMO system consists of resistors, capacitors, and MOSFETs with an $I-V$ characteristic $i = Qv^2$, where Q is a constant.

The circuit can be described by the following state-space formulation

$$C\dot{v} + Gv + Mv \otimes v = bu, \tag{13}$$

where $v \in \mathbb{R}^3$ is the vector of node voltages, u is the input to the system, b is the input matrix, $C \in \mathbb{R}^{3\times3}$ is the capacitance matrix, $G \in \mathbb{R}^{3\times3}$ and $M \in \mathbb{R}^{3\times9}$ are the first- and second-order conductance matrices, respectively. In this circuit,

$$C = \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & C_3 \end{bmatrix},$$

$$G = \begin{bmatrix} G_1 & 0 & 0 \\ -G_2 & G_2+G_3 & -G_3 \\ 0 & -G_3 & G_3 \end{bmatrix},$$

$$M = \begin{bmatrix} M_1+M_2 & -M_2 & 0 \\ -M_2 & M_2 & 0 \\ 0 & 0 & 0 \\ 0 & M_2 & -M_2 \\ -M_3 & -M_2-M_3 & -M_3 \\ 0 & M_3 & M_3 \\ 0 & 0 & 0 \\ 0 & M_3 & M_3 \\ 0 & -M_3 & -M_3 \end{bmatrix}^T,$$

$$b = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T.$$

The input is a current source signal defined by $u = i(t) = I_0 e^{-at}$ whose Laplace transform is $I(s) = \frac{I_0}{s+a}$.

According to [12, 13], given the input stimulus $i(t)$, the output voltage response $v(t)$ can be expanded into a Volterra series

$$v(t) = \sum_{n=1}^{\infty} \int \cdots \int h_n(\tau_1, \ldots, \tau_n) \cdot i(t-\tau_1) \cdots i(t-\tau_n) d\tau_1 \cdots d\tau_n, \tag{14}$$

where $h_n(\tau_1, \tau_2, \ldots, \tau_n)$ is the generalized impulse response of an n-th order system, which is called the n-th order Volterra kernels. In the frequency domain, by taking multidimensional Laplace transform we get

$$V_n(s_1, s_2, \ldots, s_n) = H_n(s_1, s_2, \ldots, s_n) I(s_1) \cdots I(s_n). \tag{15}$$

If the n-th order transfer function $H_n(s_1, s_2, \ldots, s_n)$ is known, the output spectrum $V_n(s_1, s_2, \ldots, s_n)$ can be expressed in terms of the input spectrum $I(s_1), I(s_2), \ldots, I(s_n)$. By inverting $V_n(s_1, s_2, \ldots, s_n)$ through our proposed fast multidimensional inverse Laplace transform approach and by letting $t_1 = t_2 = \cdots = t_n = t$, $v_n(t)$ can be obtained for the given $i(t)$. Summation of $v_n(t)$ at various orders gives the overall output response $v(t)$. The method is applicable to any nonlinear systems. However, simulating strongly nonlinear systems may be inefficient due to the large number of Volterra terms.

If we consider the first three orders only, which is usually sufficient for weakly nonlinear analog/RF circuit analysis, using a harmonic input method [14, 15], we can get the first three order transfer functions

$$\begin{aligned} H_1(s) &= (G+sC)^{-1}b, \\ H_2(s_1, s_2) &= -(G+\bar{s}C)^{-1} M \overline{H_1(s_1) \otimes H_1(s_2)}, \end{aligned}$$

where $\bar{s} = s_1 + s_2$, $\overline{H_1(s_1) \otimes H_1(s_2)} = \frac{1}{2}(H_1(s_1) \otimes H_1(s_2) + H_1(s_2) \otimes H_1(s_1))$ and

$$H_3(s_1, s_2, s_3) = -2(G+\bar{s}C)^{-1} M \overline{H(s_1), \otimes H_2(s_2, s_3)},$$

where $\bar{s} = s_1 + s_2 + s_3$ and $\overline{H(s_1) \otimes H_2(s_2, s_3)} = \frac{1}{6}(H(s_1) \otimes H_2(s_2, s_3) + H(s_2) \otimes H_2(s_1, s_3) + H(s_3) \otimes H_2(s_1, s_2) + H_2(s_1, s_2) \otimes H(s_3) + H_2(s_1, s_3) \otimes H(s_2) + H_2(s_2, s_3) \otimes H(s_1))$.

Thus the time response can be written as

$$\begin{aligned} v(t) = \; & v_1(t) + v_2(t) + v_3(t) \\ = \; & {L_1}^{-1}(V_1(s_1)) + {L_2}^{-1}(V_2(s_1, s_2)) \\ & + {L_3}^{-1}(V_3(s_1, s_2, s_3)). \end{aligned}$$

We simulate the circuit example with the Laguerre method by using one-core CPU without AMR acceleration technique (1CPU), one-core CPU with AMR acceleration (1CPUAMR), quadcore CPU with AMR acceleration (4CPUAMR), and GPU without AMR acceleration (GPU). Fig. 2 shows the first three orders of Volterra series of the three nodal voltages computed on GPU platform and the sum compared with a MATLAB ODE45 solution when $C = 1mF$, $G_1 = G_2 = G_3 = 10mS$, $M_1 = M_2 = M_3 = 0.01mS$, $I_0 = 1mA$, and $a = 5$. Table III compares the computation times with the number of time steps being 1000 and 5000 when $a = 0$, $a = 3$, $a = 5$ and $a = 8$. The computation time is the sum of the computation times for calculating each of the three Volterra terms, and it include the whole time from calculating the transfer functions $H_n(s_1, s_2, \ldots, s_n)$ to inverse Laplace transform. The Numerical results show that our proposed AMR technique can speed up the basic Laguerre method, and when implemented with a parallel routine in AMR on quadcore workstation, speedup becomes more significant. Quadcore CPU with AMR acceleration can speed up one-core CPU with AMR acceleration computation by up to 2.4 times and the one-core CPU calculation without AMR by up to 17 times, GPU can provide a speedup of more than 15 times. The table does not include GPU with AMR result because the constraint of condition checking in GPU prevents it from using AMR. We have tested GPU with AMR on the $a = 0$ case and results show that it slows down the calculation by 3.1 times compared with GPU without AMR. This

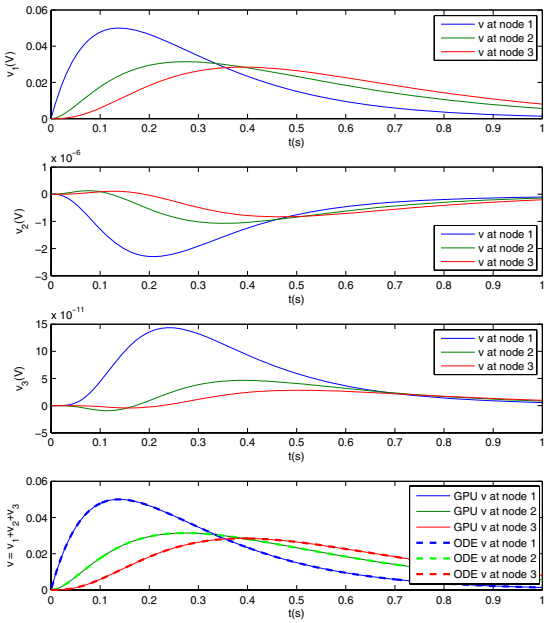

Fig. 2. The three nodal voltages in the circuit in Fig. 1 computed on GPU platform when $a = 5$. $v_1(V)$, $v_2(V)$ and $v_3(V)$ are calculated from the 1st-order, 2nd-order and 3rd-order Volterra kernels respectively, $v = v_1 + v_2 + v_3(V)$ is the overall output response, compared with an ODE45 solution.

TABLE III
COMPUTATION TIME(S) OF 1CPU, 1CPUAMR, 4CPUAMR, AND GPU WITH NUMBERS OF TIME STEPS BEING 1000 AND 5000 IN COMPUTING $v(t)$ UP TO THREE VOLTERRA TERMS WHEN $a = 0$, $a = 3$, $a = 5$ AND $a = 8$

	Steps	1CPU	1CPUAMR	4CPUAMR	GPU
a=0	(1000)	10.4 s	4.76 s	3.47 s	0.555 s
	(5000)	51.7 s	8.91 s	5.14 s	2.483 s
a=3	(1000)	10.4 s	3.71 s	2.40 s	0.536 s
	(5000)	52.1 s	7.92 s	4.13 s	2.48 s
a=5	(1000)	10.4 s	3.49 s	1.90 s	0.541 s
	(5000)	51.5 s	7.72 s	3.47 s	2.46 s
a=8	(1000)	10.4 s	2.91 s	1.41 s	0.562 s
	(5000)	51.8 s	7.08 s	2.89 s	2.45 s

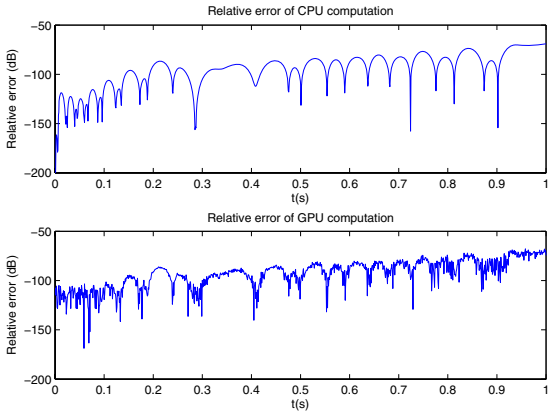

Fig. 3. Relative errors of CPU and GPU computation solution of the circuit in Fig. 1 when $a = 5$.

is because the GPU speedup is already significant, the condition check in AMR incurs a large overhead for GPU architecture that may hamper its otherwise high data computation efficiency. Our on-going work is to study smart CPU/GPU co-simulation to move the sequential parts to CPU whereas parallelizable part to GPU automatically instead of manually, eventually leading to ultimate speedup.

The relative errors of the above CPU and GPU solution up to three Volterra terms compared with a MATLAB ODE45 solution when $a = 5$ are shown in Fig. 3. We can see that accurate results can be obtained by the fast inverse Laguerre-based Volterra analysis.

Example 2

For more general and large scale examples, next, we consider a circuit example of an RC ladder shown in Fig. 4. A quadratic nonlinearity is introduced to the circuit due to nonlinear resistors to the ground at each node $i_n(v) = g \cdot v^2$, where $g = 0.01mS$. The nonlinear circuit can be described by a state-space formulation with the same format as in (13), with $v \in \mathbb{R}^n$ the vector of node voltages, u the input to the system, $b \in \mathbb{R}^{n\times 1}$ the input matrix, $C \in \mathbb{R}^{n\times n}$ the capacitance matrix, $G \in \mathbb{R}^{n\times n}$ and $M \in \mathbb{R}^{n\times n^2}$ the first- and second-order conductance matrices, respectively. n is the number of nodes. Similar to the first example, up to three orders of Volterra series are computed. Fig. 5 shows the first five nodal voltages in the circuit in Fig. 4 when $n = 50$, computed using one-core CPU, GPU with our proposed method, and MATLAB ODE45, when $C = 1mF$, $r = 1\Omega$, $g = 0.1mS$, and $u = i(t) = e^{-5t}$.

We simulate the circuit example of various sizes (when $n = 10$, $n = 50$, and $n = 100$) using one-core CPU without AMR acceleration technique (1CPU), one-core CPU with AMR acceleration technique (1CPUAMR), quadcore CPU with AMR acceleration technique (4CPUAMR), and GPU without AMR. Again, GPU with AMR is not included because the constraint of condition checking in GPU highly increases the whole runtime. The computation times of calculating the output voltages at the first ten nodes and relative errors compared with the ODE45 solution are shown in Table IV. The computation times in the table are the sum of the computation times for calculating each of the first two Volterra terms, and include the whole time from calculating the transfer functions $H_n(s_1, s_2, \ldots, s_n)$ to inverse Laplace Transform. The table does not include the third-order Volterra term runtime because it has little contribution to the output response and the first two orders have already

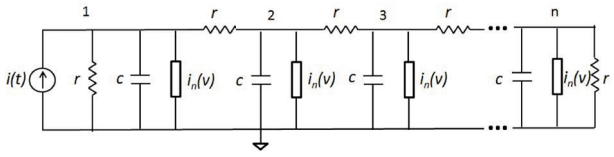

Fig. 4. Example of a large scale circuit with quadratic nonlinearity.

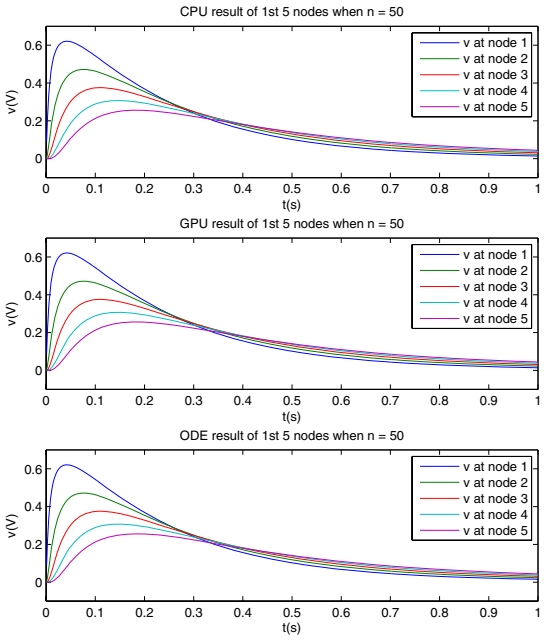

Fig. 5. The first five nodal voltages in the circuit in Fig. 4 with a size of 50 ($n = 50$), computed using one-core CPU, GPU with our proposed method, and MATLAB ODE45.

shown significant speedup. The number of Volterra terms that need to be calculated increases when coefficient of the second-order term g increases. In this test example the number of time steps is 5000, simulation time is 0-5 s. It can be seen from the table that high accuracy and significant speedup can be achieved for simulating large nonlinear circuits by utilizing GPU.

V. Conclusion

An efficient frequency-domain parallelizable simulation method has been proposed for simulating nonlinear circuits. The proposed algorithm employs numerical multidimensional inverse Laplace transform based on Laguerre function expansion. By using a proposed parallel routine and AMR technique, the method enjoys both good accuracy and high efficiency.

TABLE IV
Computation Time(s) and Relative Errors of 1CPU, 4CPU, GPU and GPU in Computing the Output Voltages at the First Ten Nodes in the Circuit in Fig. 4 when $n = 10$, $n = 50$, and $n = 100$. n is the Circuit Size (Number of Nodes).

		1CPU	1CPUAMR	4CPUAMR	GPU
n=10	runtime	34.9 s	8.23 s	6.21 s	1.02 s
	error(%)	0.0622	0.109	0.108	0.0625
n=50	runtime	355 s	84.0 s	29.2 s	6.61 s
	error(%)	0.0819	0.143	0.143	0.0818
n=100	runtime	1010 s	221 s	69.1 s	13.5 s
	error(%)	0.0791	0.145	0.140	0.0808

Acknowledgments

This work is supported in part by the Hong Kong Research Grant Council under GRF Project 718711E, and in part by the University Research Committee of the University of Hong Kong.

References

[1] Abate, J. and Choudhury, G.L. and Whitt, W., "Numerical inversion of multidimensional Laplace transforms by the Laguerre method," *Performance Evaluation*, vol. 31, pp. 229–243, 1998.

[2] Roychowdhury, J., "Analyzing circuits with widely separated time scales using numerical PDE methods," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 48, pp. 578–594, 2001.

[3] H. Liu, F. Shi, Y. Wang and N. Wong, "Frequency-domain transient analysis of multitime partial differential equation systems" in Proc. Very Large Scale Integration (VLSI-SoC), pp. 160-163, Oct 2011.

[4] Brancik, L., *Numerical inversion of 3D Laplace transforms for weakly nonlinear systems solution,* Radioelektronika (RADIOELEKTRONIKA), 20th International Conference, 2010.

[5] Hwang, C. and Guo, T.Y. and Shih, Y.P., "Numerical inversion of multidimensional Laplace transforms via block-pulse functions," *Control Theory and Applications, IEEE Proceedings D*, vol. 130, pp. 250–254, 1983.

[6] Nastov, O. and Telichevesky, R. and Kundert, K. and White, J., "Fundamentals of fast simulation algorithms for RF circuits," *Proceedings of the IEEE*, vol. 95, pp. 600–621, 2007.

[7] Joyati, D. and Narayan, C., "Associated transforms for solution of nonlinear equations," *International Journal of Mathematics and Mathematical Sciences*, vol. 14, pp. 177–190, 1991.

[8] Valkó, P.P. and Abate, J, "Numerical inversion of 2-D Laplace transforms applied to fractional diffusion equations," *Applied numerical mathematics*, vol. 53, pp. 73–88, 2005.

[9] Abate, J. and Choudhury, G.L. and Whitt, W., "On the Laguerre method for numerically inverting Laplace transforms," *INFORMS Journal on Computing*, vol. 8, pp. 413–427, 1996.

[10] Choudhury, G.L. and Lucantoni, D.M. and Whitt, W., "Multidimensional transform inversion with applications to the transient M/G/1 queue," *The Annals of Applied Probability*, pp. 719–740, 1994.

[11] Abate, J. and Choudhury, G.L. and Whitt, W., "Adaptive mesh refinement for hyperbolic partial differential equations," *Journal of Computational Physics*, vol. 53, pp. 484–512, 1984.

[12] Wilson J. Rugh, *Nonlinear System Theory: The Volterra/Weiner,* The Johns Hopkins University Press, 1989.

[13] Schetzen, M., "The Volterra and Wiener Theories of Nonlinear Systems," *John Wiley & Sons*, 1980.

[14] Li, P. and Pileggi, L.T., "Compact reduced-order modeling of weakly nonlinear analog and rf circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, pp. 184–203, 2005.

[15] Bussgang, J.J. and Ehrman, L. and Graham, J.W., "Analysis of nonlinear systems with multiple inputs," *Proceedings of the IEEE*, vol. 62, pp. 1088–1119, 1974.